POLITECNICO DI TORINO
Repository ISTITUZIONALE

CHAMALEONET: Programmable Passive Probe for Enhanced Visibility on Erroneous Traffic

Original
CHAMALEONET: Programmable Passive Probe for Enhanced Visibility on Erroneous Traffic / Wang, Z., Cornacchia, A., Bianco, A., Drago, I., Giaccone, P., Jiang, D., Mellia, M.. - In: IEEE TRANSACTIONS ON NETWORKING. - ISSN 2998-4157. - 34:(2026), pp. 6812-6826. [10.1109/TON.2026.3723852]

Availability:
This version is available at: 11583/3014589 since: 2026-08-18T14:12:46Z

Publisher:
IEEE

Published
DOI:10.1109/TON.2026.3723852

Terms of use:

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

Publisher copyright

(Article begins on next page)

19 November 2026

CHAMALEONET: Programmable Passive Probe for Enhanced Visibility on Erroneous Traffic

Zhihao Wang, Alessandro Cornacchia, Andrea Bianco, *Senior Member, IEEE*, Idilio Drago, Paolo Giaccone, *Senior Member, IEEE*, Dingde Jiang, *Member, IEEE*, and Marco Mellia, *Fellow, IEEE*

***Abstract*—Traffic visibility remains a key component for management and security operations. Observing erroneous traffic, i.e., *unanswered requests or error messages*, is fundamental to detecting misconfiguration, temporary failures or attacks. CHAMALEONET transforms any production network into a transparent monitor to let administrators collect such erroneous traffic. CHAMALEONET is programmed to ignore well-formed traffic and record only erroneous packets, including those generated by misconfigured or infected internal hosts, and those sent by external actors that scan for services. Engineering such a system poses several challenges, from scalability to privacy. Leveraging the Software-Defined Networking (SDN) paradigm, CHAMALEONET processes the humongous amount of traffic flowing through the network border and focuses on erroneous packets only, lowering the pressure on the collection system. Moreover, it offers traffic anonymisation to conform to privacy regulations. CHAMALEONET enables the seamless integration with active deceptive systems like honeypots that can impersonate hosts/ports/services and engage with senders. In an operational scenario, we show that the SDN in-hardware filtering reduces the traffic to the controller by 90%, resulting in a scalable solution, which we offer as open source. Simple statistical analytics unveil the precious information carried by erroneous traffic. We discover internal misconfigured and infected hosts, identify temporary failures, and show enhanced visibility on attackers' scanning activities that look for vulnerable services.**

***Index Terms*—Traffic visibility, network monitoring, software-defined networking, programmable data plane.**

Received 6 September 2025; revised 22 February 2026 and 26 June 2026; accepted 16 July 2026; approved by IEEE TRANSACTIONS ON NETWORKING Editor S. Alouf. Date of publication 14 August 2026; date of current version 3 September 2026. This work was supported in part by Project SERICS (SEcurity and RIghts In the CyberSpace; PE00000014) under the MUR National Recovery and Resilience Plan funded by the European Union—NextGenerationEU and in part by the PRIN (Progetti di Rilevante Interesse Nazionale) project ACRE (AI-Based Causality and Reasoning for Deceptive Assets; 2022EP2L7H). *(Corresponding author: Zhihao Wang.)*

Zhihao Wang is with the University of Electronic Science and Technology of China, Chengdu 611731, China, and also with the Department of Control and Computer Engineering, Politecnico di Torino, 10129 Turin, Italy (e-mail: zhihao.wang@polito.it).

Alessandro Cornacchia is with the Computer, Electrical and Mathematical Sciences and Engineering Division, King Abdullah University of Science and Technology, Thuwal 23955, Saudi Arabia (e-mail: alessandro.cornacchia@kaust.edu.sa).

Andrea Bianco and Paolo Giaccone are with the Department of Electronics and Telecommunications, Politecnico di Torino, 10129 Turin, Italy (e-mail: andrea.bianco@polito.it; paolo.giaccone@polito.it).

Idilio Drago is with the Department of Computer Science, Universitá di Torino, 10149 Turin, Italy (e-mail: idilio.drago@unito.it).

Dingde Jiang is with the School of Information and Communication Engineering, University of Electronic Science and Technology of China, Chengdu 611731, China (e-mail: jiangdd@uestc.edu.cn).

Marco Mellia is with the Department of Control and Computer Engineering, Politecnico di Torino, 10129 Turin, Italy (e-mail: marco.mellia@polito.it).

Digital Object Identifier 10.1109/TON.2026.3723852

I. INTRODUCTION

NETWORK monitoring has always been the first step to implement network management or security policy. Some monitoring systems offer visibility on regular traffic to derive statistics on performance and application usage, e.g., flow loggers [1], [2], [3]; some systems focus on protecting the network infrastructure, e.g., Intrusion Detection/Prevention Systems (IDPSs) [4]; some collect unsolicited traffic, e.g., network telescopes [5], [6].

Yet live networks contain well-formed flows, i.e., requests that get answered by the destination, and flows that remain unanswered or generate error messages at the network layer. In this paper we focus on the latter, which we term *erroneous traffic*. Such traffic arises when clients attempt to connect to offline (or firewalled) systems, or try to reach unintended or unavailable servers (possibly caused by routing anomalies). At last, malicious actors scanning networks and services generate erroneous traffic.

Prior work [7], [8], [9], [10] has shown erroneous traffic reveals interesting insights. For example, [8] shows that unsolicited traffic in large Content Delivery Networks (CDNs) is not uniformly random: over 30% of scans are localized, targeting few address regions. Similarly, DScope [7] reports up to $450\times$ more scan traffic than expected under random scanning. These results indicate that erroneous traffic in production networks reflects systematic targeting and structural bias rather than background noise. This motivates detecting and collecting such traffic in production environments to study targeting effects and distribution patterns. Our previous work [11] also reveals the value of outbound erroneous traffic in identifying signals of misconfiguration, transient failures, or attacks.

In this paper, we design a passive traffic collector explicitly targeted to erroneous traffic. We identify three key requirements that an ideal collector should satisfy: *R1 Completeness:* the collector should provide full coverage of erroneous traffic, even when error status becomes evident only at runtime; *R2 Transparency:* the collector must operate on live address space, coexisting with legitimate address assignments and active services without requiring dedicated dark subnets; *R3 Data minimisation:* regulatory and institutional policies [12], [13] require that capture focuses on traffic relevant to security or operations, avoiding bulk logging of legitimate sessions.

We argue that none of the existing monitors jointly satisfies the three requirements. Traditional network telescopes record all packets destined to an *unused subnet*, i.e., where

© 2026 The Authors. This work is licensed under a Creative Commons Attribution 4.0 License.
For more information, see https://creativecommons.org/licenses/by/4.0/

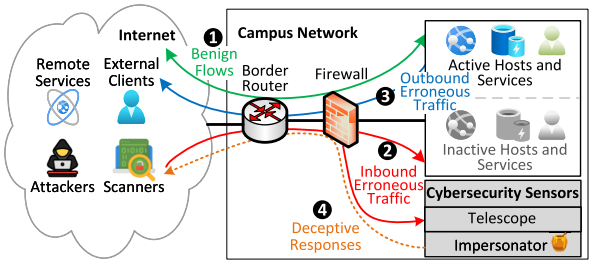


Fig. 1. Concept of erroneous traffic and scope of CHAMALEONET in a campus network, highlighting the actors and traffic types.

no production host is connected, and thus are inherently designed to capture erroneous traffic. However, they present two limitations. First, because telescopes operate on reserved and unassigned address spaces, devoting precious IP subnets solely for data collection, they cannot operate on live address space, violating R2. Flexible telescopes [7], [8], [9] exist, but still require addresses to remain unassigned during collection. Second, telescopes only capture traffic directed to the monitored dark space (e.g., inbound Internet radiation [5]), missing erroneous traffic directed to live hosts and all outbound erroneous traffic generated by internal hosts, thus violating R1. Traditional flow loggers operate on live networks but provide only statistical summaries of flows, without distinguishing erroneous traffic. Therefore, they violate R1 (e.g., due to sampling) and R3. Finally, Intrusion Detection/Prevention Systems (IDPSs) inspect packets against predefined rules or known signatures, falling short of R1 when erroneous traffic is not known a priori, and of R3 when bulk logging is enabled to capture unknown traffic. Although not designed to collect erroneous traffic, IDPSs can, in principle, be extended to track it by maintaining per-flow states (e.g., triggering events for new requests and timeouts [14]), since they inherently support stateful processing. However, stateful operation is costly, and their support for raw packet logging is limited: the scale of typical data rates challenges the usage of IDPS to log erroneous traffic [1], [3], [15].

To close this gap, we propose CHAMALEONET, a flexible monitoring system that transforms any production network into a programmable probe for erroneous traffic. Unlike traditional network telescopes, CHAMALEONET avoids reserving IP addresses for monitoring purposes and dynamically detects erroneous traffic exchanged by services in an operational, live network, and opportunistically collects it.

Fig. 1 depicts the concept of erroneous traffic and the scope of CHAMALEONET. CHAMALEONET ignores all regular traffic (green arrow ❶), and only collects those packets for which no valid response is observed (arrows ❷ ❸). More specifically, CHAMALEONET logs the erroneous traffic generated by *external hosts* (inbound erroneous traffic ❷), offering visibility on Internet radiation and attackers' scanning activities, and enabling the administrators to engage with scanners by selectively enabling impersonators. Likewise, CHAMALEONET logs requests originated by *internal hosts* (outbound erroneous traffic ❸), unveiling issues with external services and offering visibility on misconfigured or infected internal hosts, thereby facilitating troubleshooting and repair. At last, CHAMALEONET can transform itself into an active impersonator (orange dashed arrow ❹), impersonating inactive hosts or services, as a honeypot system.

Engineering CHAMALEONET requires ingenuity. We leverage Software-Defined Networking (SDN) and programmable switches to transform a live network into a flexible monitor. To identify erroneous traffic (❷ and ❸), CHAMALEONET tracks unanswered requests or error messages destined to addresses assigned to unreachable internal hosts. Once identified, CHAMALEONET steers erroneous traffic to back-end cybersecurity sensors, whether they are simple passive collectors or active impersonators that can impersonate the destination. The SDN architecture naturally maps to these design requirements. The controller handles the stateful detection logic that identifies non-responding services. Meanwhile, programmable switches perform line-rate filtering of benign traffic, shielding the controller from the full traffic volume. At the same time, network programmability enables flexible steering of erroneous traffic towards cybersecurity sensors, and offloads privacy-aware packet processing functionalities (e.g., packet anonymisers) to address regulatory requirements to the network data plane.

In summary, CHAMALEONET transforms any network into a flexible and transparent monitor for erroneous traffic, enabling visibility on a wide range of management and cybersecurity events coming from external and internal hosts. We deployed CHAMALEONET at our University campus network for over eight months, and report on the key findings. We offer CHAMALEONET to the community as open source.[1]

Our contributions are as follows: (i) We introduce the concept of erroneous traffic and design CHAMALEONET, a system that transforms any private network into a flexible monitor of erroneous traffic. (ii) We implement CHAMALEONET atop SDN principles to transparently filter, anonymise, and steer traffic, respecting users' privacy. Offloading the filtering of regular traffic to the switch reduces the controller load by 90%. (iii) CHAMALEONET improves visibility on external Internet radiation while empowering the administrators to engage with scanners by selectively enabling impersonators. (iv) Simple statistical analysis of the erroneous traffic patterns expose suspicious internal hosts, misconfigured systems, and routing issues, which would otherwise go unnoticed.

Next, Sec. II presents the related work. Sec. III presents the design and architecture of CHAMALEONET. Sec. IV describes its implementation. Sec. V reports on its deployment and key findings, while Sec. VI evaluates its performance in a controlled environment. Sec. VII outlines strategies to improve the architecture. We discuss the ethical considerations in Sec. VIII and conclude the paper in Sec. IX.

II. RELATED WORK

A. Passive Measurement Mechanisms

Several mechanisms exist to passively monitor traffic, from simple network telescopes to IDS/IPS or IDPS. Telescopes collect packets destined to dark subnets with no active hosts. The collected Internet background radiation [5], [6] gives visibility on Internet scanners, botnets, coordinated attacks, failures, routing issues and even censorship policies [16], [17].

[1] https://github.com/zhihao1998/ChamaleoNet

TABLE I
SCOPE OF CHAMALEONET

	Traffic	Focus	Pkt	Flow	Alert	Reply	Rules
Telescope	In	Dark	✓				static
Reactive Telescope	In	Dark	✓			✓	static
Flow monitor	In/Out	All		✓			static
IDPS	In/Out	Malicious	✓	✓	✓		static
CHAMALEONET	In/Out	Erroneous	✓			✓	dynamic

Telescopes ignore regular and some erroneous traffic and waste resources due to the need to leave unused large ranges of IP addresses [18].

Recent works have explored flexible telescope designs. In [8] authors leveraged CDN infrastructure to study unsolicited traffic reaching production servers, collecting packets blocked at CDN firewalls. Attracted by live CDN nodes, attackers send a variety of packets that are not observed in telescopes. Similarly, DScope [7] places telescopes in cloud data centers. It continually and dynamically allocates new IP addresses from the Amazon Web Services, retaining them for 10 minutes to collect inbound TCP connection attempts. Meta-telescope [9] infers unused Internet address blocks and captures the corresponding traffic without requiring administrative control over those blocks.

These designs require that addresses remain *unassigned/unused* when used for collecting unsolicited traffic. As such, they lack visibility into: *i)* addresses that are temporarily offline but not yet incorporated into the pool, and *ii)* addresses with only a subset of ports active. In contrast, CHAMALEONET immediately gains visibility of a host once it goes dark and continuously monitors the unused ports of *active* hosts.

Reactive telescopes such as Spoki [19] and others [20] augment telescopes with the ability to respond to incoming requests through simple impersonators, e.g., opening the TCP connection to capture the first payload.

On the other extreme, IDPSes, e.g., Snort, Suricata and Zeek, analyse live traffic in real time and block malicious or suspicious activities, eventually alerting administrators [4]. IDPSes react to attacks and system abuses, using known signatures or anomaly detection. They offer the ability to collect packet traces, selectively logging only packets causing the incident [21] or logging all packets.

In between, flow loggers offer visibility on operational traffic [1], [2], [3], processing packets in real-time to export flow-level statistics, but provide only limited packet capture capabilities, typically for traffic samples or specific protocols.

Table I summarises the CHAMALEONET scope. All platforms process incoming traffic. Flow monitors, IDPSes and CHAMALEONET offer visibility on traffic initiated by internal hosts (outgoing traffic). Telescopes focus on dark traffic, flow monitors summarize all traffic, while IDPSes are usually set to look for malicious traffic. Telescopes capture packets, while flow monitors log information at the flow level. IDPSes produce alerts based on rules, and have some packet capture abilities. Only reactive telescopes and CHAMALEONET support backend impersonators that can actively respond to incoming requests, such as completing the TCP handshake.

B. Monitoring and Defence Mechanisms in SDN

The authors of [22] provide a comprehensive taxonomy of past work that leveraged the use of SDN as a defence mechanism. Closest to our work are those related to honeypot systems. The study in [23] introduced a honeynet based on SDN, akin to CHAMALEONET, which duplicates traffic to the honey-network for further examination. In [24], authors proposed a solution where the controller inspects suspicious traffic packets and redirects it to honeypots. None of these works can dynamically filter well-formed traffic at runtime and redirect only *erroneous* traffic to collector systems.

C. Stateful Traffic Processing With Programmable Switches

CHAMALEONET enables stateful processing of incoming traffic by leveraging the SDN switch and virtualized network functions. Some past works have focused on implementing stateful processing only in the switch, enabled by Programming Protocol-independent Packet Processors (P4) [25]. Reference [26] proposed a reactive traffic control application that redirects traffic in real-time to a traffic classification engine, entirely within the SDN switch. Reference [27] leverages a stateful approach directly in data plane to detect TCP SYN flood attacks. Reference [28] uses SDN switch to mirror traffic to IDS engines, where the controller may instruct the switch to block malicious flows. Unlike CHAMALEONET, it neither buffers packets while the decision is being made, including the critical first packet, nor considers erroneous or internally generated traffic.

Summary. CHAMALEONET is the only system that targets erroneous traffic. It places itself between passive telescopes and advanced flow monitors and IDPSes, offering visibility on both external and internal erroneous traffic. CHAMALEONET differs from previous works in several key aspects: 1) focuses on erroneous traffic, which previous systems largely ignore; 2) leverages SDN programmability to detect erroneous traffic through dynamic rules; 3) provides visibility into both incoming and outgoing traffic; 4) operates on live networks and transparently treats unused addresses–including temporarily offline hosts–as dark space; 5) extends the scope of dark space to the service level, i.e., unused ports on otherwise active servers; 6) logs traffic at the packet level while preserving privacy; 7) can selectively steer traffic to impersonators which can impersonate inactive hosts or specific services.

III. SYSTEM DESIGN

Fig. 2 shows an overview of CHAMALEONET. We design it as a pluggable network system (blue box) with minimal configuration. Connected to the campus network border router(s), CHAMALEONET observes all traffic entering/leaving the network and forwards only the erroneous portion to the collectors, optionally letting collectors impersonate the destination host or service. Upon observing the first packet of a new flow,[2] it must determine whether the destination service is active, and eventually forward traffic destined to inactive services or hosts to the backend. CHAMALEONET must not cause any

[2]We define a flow via the usual 5-tuple, up to the transport layer. Flows are bi-directional: packets matching the same 5-tuple belong to the same flow, regardless of their direction.

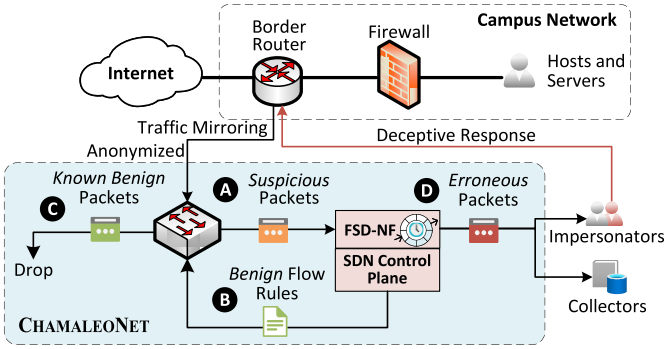


Fig. 2. Overview of ChamaleoNet architecture.

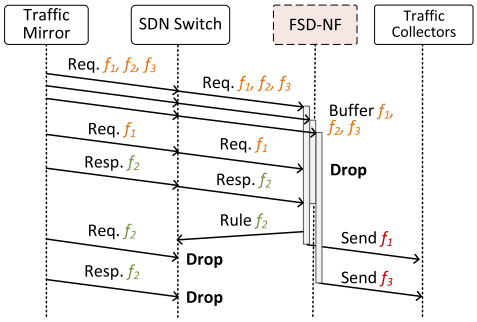


Fig. 3. ChamaleoNet detecting two flows with erroneous packets (f_1, f_3) and one benign flow (f_2).

disruption or interference to the active services, nor forward production traffic payload to the collectors. Hence, we design the system to be *transparent* to services and benign traffic.

A. ChamaleoNet Architecture

ChamaleoNet leverages the SDN paradigm to separate the data plane, responsible for line-rate packet filtering, from the control plane, which manages the stateful logic to distinguish erroneous from benign traffic.

We here assume an out-of-path deployment (see Sec. VII for a discussion of the in-path alternative). All ingress and egress traffic, destined to and coming from the campus network, is mirrored to an SDN switch via either span ports or physical layer splitters. In case of multiple upstream providers, we assume all traffic is forwarded to handle asymmetric routing.

The switch applies dynamic per-service filters. When a packet arrives and does not match any filter, the switch forwards it to the SDN controller. Such a packet may either belong to a new legitimate flow or be potentially erroneous. Since both cases are plausible a priori, initially we denote these packets as *suspicious packets* (event Ⓐ in Fig. 2).

ChamaleoNet is designed to minimize logging of traffic destined to active services. To achieve this, a dedicated *Flow State Detection Network Function (FSD-NF)* determines whether a suspicious packet is *answered* (the service is active) or *unanswered* (the service is temporarily inactive). FSD-NF buffers the suspicious packet for a pre-defined DeTection (DT) timeout during which it waits for a response packet of the same flow. Two cases can occur:

1) A response packet is received within the DT timeout: the service is deemed *active* and the suspicious packet (and following packets) are part of a legitimate flow. The controller drops the packet and installs a flow rule on the SDN switch (Ⓑ) to drop all matching packets. The SDN switch will discard any subsequent packet of the benign flow, avoiding overflowing the controller (Ⓒ).
2) The DT timeout expires: the service is deemed *inactive* and the suspicious packet is classified as *erroneous*. The controller forwards the original packet buffered at the FSD-NF to the cybersecurity collectors (Ⓓ), which may log or eventually reply to such packets acting as a honeypot.

Albeit easy to implement, ChamaleoNet treats ICMP unreachable errors as erroneous rather than valid responses.

By observing all initial flow packets and responses, ChamaleoNet can keep track of which internal hosts are *alive*: Whenever the sender IP address corresponds to an internal host, ChamaleoNet marks the sending host as alive. This allows it to distinguish cases where external requests go unanswered because the destination is not alive (or dark, as in the case for telescopes) or present but refusing to answer (e.g., firewalled service or rebooting hosts). ChamaleoNet can take decisions on whether to respond to a packet, log or ignore it accordingly.

B. Running Example

The time diagram in Fig. 3 shows an example of how ChamaleoNet works. The switch receives traffic from the traffic mirror. It receives three packets belonging to three different flows (f_1, f_2, f_3). No rule matches these packets, so the switch forwards them to the FSD-NF, which buffers them and sets a DT timer to wait for more packets of the same flow to eventually arrive. Next, the FSD-NF observes a second request packet belonging to f_1 (e.g., a retransmission from the same sender), and drops it. Eventually, this second packet could be stored at the FSD-NF and sent to the cybersecurity collector too. Then, a response packet for f_2 arrives and is forwarded to FSD-NF, which deems the service and the host as active. FSD-NF marks the flow f_2 as "benign" and installs a rule in the switch to drop all subsequent packets belonging to f_2. In the meantime, FSD-NF drops all f_2 packets that may still be forwarded by the switch. At last, f_1 and f_3 DTs expire. The FSD-NF sets the corresponding buffered packets as "erroneous" and sends them to the collectors, removing any internal state related to them.

C. Assumptions and Threat Model

Assumptions. We make the following assumptions: *1) Unified visibility:* the monitored network has unified borders and all ingress/egress traffic is mirrored to ChamaleoNet (out-of-path); *2) Asymmetric routing:* in multi-homing settings, traffic is mirrored/forwarded such that asymmetric return paths remain visible. Note that asymmetric routing is an open challenge for such monitoring systems, e.g., in Meta-TeleScope [9]; *3) Timing consistency:* packet timestamps and ordering at the monitoring vantage point are sufficiently accurate for enforcing the observation window; *4) No inbound spoofing:* border devices are trusted and enables common filtering that

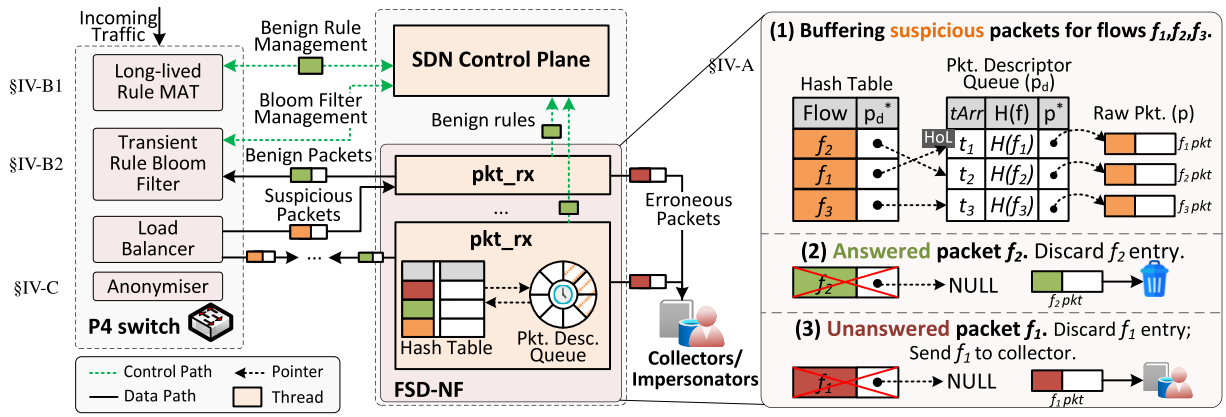


Fig. 4. Implementation of ChamaleoNet and its user-space packet processing workflow.

prevents external hosts from spoofing internal source IPs (e.g., BCP38 [29]). *5) Dynamic SDN-like rule support in the data plane:* the switch component—whether a physical SDN/P4 switch, a SmartNIC, or an extended Berkeley Packet Filter (eBPF)/eXpress Data Path (XDP) program—accepts dynamic flow rules and forwards unmatched packets to the controller.

Threat model. ChamaleoNet captures interactions in which requests reach routable addresses, but no observable reverse-direction response is detected within a defined observation window. Such interactions arise when endpoints are inactive, filtered, unreachable at the service layer, misconfigured, or non-responsive. The threat model, therefore, includes network activities interacting with non-responsive endpoints. Examples include scanning, probing, exploitation attempts, and unintended communications. These activities may be malicious or benign in intent. The defining property is the absence of an observable response within the observation window, not the intent of the sender. The goal of ChamaleoNet is to identify and analyse such interactions without disrupting normal traffic forwarding.

Monitoring blind spots and evasion. The threat model is limited to L3/L4 response observability within a configurable observation window. Interactions that successfully establish bidirectional communication within the observation window fall outside the erroneous-traffic model. An attacker could evade detection by injecting crafted reverse-direction packets that traverse the monitored border within the DT timeout. This requires coordinated control over the reverse path (e.g., control of both endpoints or reverse-path injection capability). In such cases, the resulting interaction becomes indistinguishable from legitimate L3/L4 communication under ChamaleoNet's observability model. Detecting such behaviours is outside the scope of ChamaleoNet and is typically delegated to other monitoring mechanisms, such as IDPSes.

IV. Implementation

Fig. 4 depicts the main components of ChamaleoNet and their interactions. We first describe the FSD-NF packet processing logic and data structures (Sec. IV-A), then present a two-tier rule management strategy for filtering benign flows at the SDN switch (Sec. IV-B), and finally highlight the traffic anonymizer (Sec. IV-C).

A. Flow State Detection Network Function (FSD-NF)

We rely on state-of-the-art per-flow monitoring based on multi-threading to implement the FSD-NF, which has been shown to cope with several tens of Gbps on off-the-shelf hardware [1]. For packet reception, ChamaleoNet uses any library that allows to capture packets. In our prototype, we use `libpcap` configured with "immediate mode", which disables packet batching and delivers packets to the FSD-NF upon arrival. Any other kernel-bypass packet I/O frameworks, such as DPDK [30], are transparent to ChamaleoNet design, and orthogonal to the scope of this work. At the FSD-NF, multiple `pkt_rx` threads process suspicious packets arriving from the switch, buffering packets and executing the FSD-NF packet processing logic described next. Suspicious packets are load-balanced across `pkt_rx` threads via hash-based selection on internal service identifiers. This preserves per-service affinity because the hashing operation is offloaded to the switch.

1) Packet Buffering and Data Structures: Fig. 4 shows the ChamaleoNet data structures and processing. There are three main data structures in FSD-NF: a hash table to store flow states; a packet descriptor queue to manage timeouts; a packet buffer to cache raw packets. The FSD-NF buffers suspicious packets and waits for a response (benign flow) or for the DT timer to expire (erroneous packet). We store packets in FSD-NF memory and track their locations using descriptors organized as a ring buffer queue. The descriptor stores also the packet arrival time *tArr* and a pointer to the entry in the flow hash table. Event (1) in Fig. 4 (right) illustrates the relationship between the data structures for packets of flow f_1, f_2 and f_3, while Algorithm. 1 shows the pseudocode. When a new packet arrives and does not match any entry in the hash table, we store it in memory, create a new packet descriptor, set the $tArr$ field to the current time, and insert the packet descriptor in the packet descriptor queue. To optimise the lookup when matching responses, we also store in the hash table a pointer to the packet descriptor position within the ring buffer.

2) Erroneous Packet Detection and DT Timer Management: ChamaleoNet detects erroneous packets by waiting for the DT timeout to expire. To avoid the overhead of managing multiple timers, one per packet, we implement a lazy timeout mechanism with a single timer. We exploit the simple observation that DT timeouts for suspicious packets expire in the same

order in which the packets are received, and manage expiration via the following polling mechanism embedded within the pkt_rx thread (ln. 20). The pkt_rx thread periodically (every P_{DT}) scans the packet descriptor queue starting from the entry with oldest $tArr$. For each entry, pkt_rx compares the current wall-clock time with *tArr*. If the elapsed time is greater than the DT timeout, it deletes the hash table entry, sends the packet to the collector and frees the corresponding packet descriptor. We stop at the first descriptor that has not exceeded the DT timeout. Event (3) in Fig. 4 (right) illustrates a packet of flow f_1 marked erroneous and sent to the collector. To limit the computation cost of this linear search operation, we allow a maximum search depth equal to $d_{\max}$ positions.[3]

3) Benign Flow Detection: If FSD-NF receives a response packet before the DT expiration, the flow is deemed *benign*—i.e., internal service is active (Sec. III-A). FSD-NF deletes the corresponding suspicious packet from the packet descriptor queue, and installs an entry in the switch to filter all future packets of this benign flow. Response packets are detected in constant time using hash table lookups, by matching the response packet's flow identifier to the corresponding suspicious packet's flow entry.

B. Filtering Rules Management

The FSD-NF dynamically installs filtering rules in the switch so that subsequent packets belonging to benign flows are dropped. Filtering rules take the form of exact-match rules in Match-Action Tables (MATs) or approximate matching in Bloom filters (BF). We refer to the mechanism as a dual-channel rule management. A *fast-path* channel quickly pushes filtering rules to the switch using in-band packets that update BFs via the switch dataplane. A *slow-path* channel manages the installation of long-lived rules for well-known flows via the control plane. The terminology "fast" and "slow" refers to the latency of rule installation in the switch, which is negligible for BFs and can be up to hundreds of milliseconds for MATs.

We describe how we balance the use of these two channels to filter benign traffic while keeping the switch rule set manageable.

1) Slow Path: Filtering Long-Lived Benign Flows With MAT Rules: In production networks, long-lived internal services (e.g., web services) always respond to inbound packets and consistently generate well-formed bidirectional traffic, i.e., benign flows. CHAMALEONET identifies long-lived benign services and installs persistent filtering rules on the P4 switch using MATs. Because of stability and predictability, flows belonging to long-lived services are ideal candidates for the slow-path channel: since MAT memory has limited capacity and longer installation latency, we reserve MAT rules for filtering stable and well-known services.

The SDN control plane manages both rule installation and removal. When pkt_rx threads identify flows belonging to these services as benign, they trigger rule installation via the SDN control plane, which installs exact-match MAT rules keyed by `(internal IP, internal port, protocol)`. Due to limited MAT memory capacity, the control plane also implements rule eviction to delete idle entries for (likely) terminated flows. We implement the entry eviction based on the idle notification mechanism provided by Intel Tofino chipset. Each MAT entry maintains a `TTL` field that decrements when no packet matches within a `Query_interval`; any matching packet resets the `TTL`. When the `TTL` reaches zero, the switch triggers an idle timeout notification to the controller, which also resets the corresponding host's aliveness status. We employ batching both for installation and removal to reduce overhead.

2) Fast Path: Filtering Transient Benign Flows With Bloom Filters: The fast-path is designed to cope with periods of high benign flow churn, where many new flows are initiated within short time intervals. In such scenarios, the switch would

[3]While this check could be done in a separate thread, it would create race conditions on the ring buffer, introducing synchronisation overheads. Experimental results confirm that it is not needed (see Sec. VI).

Algorithm 1 pkt_rx and DT Timer Management in FSD-NF

Require: Packet descriptor queue Q, hash table H, DT timeout, polling period P_{DT}, max search depth $d_{\max}$
Ensure: Suspicious packets p belonging to flow f forwarded to collector, expired descriptors freed

On new suspicious packet arrival p:
1: **if** p matches an existing entry in H **then**
2: **if** $H_f.tResp \neq$ `null` **then** ▷install pending
3: Drop p and **return**
4: **else if** $p.dir = H_f.dir$ **then** ▷same-direction duplicate
5: Drop p and **return**
6: **else** ▷reverse-direction (response) packet
7: Install a benign rule to the switch
8: $p_d^* \leftarrow$ `null`
9: Delete H_f from hash table H
10: **end if**
11: **else** ▷suspicious first packet
12: Create packet descriptor p_d
13: $p_d.tArr \leftarrow$ `currentTime()`
14: Buffer packet p
15: $p_d.p^* \leftarrow addr(p)$ ▷pointer to buffered packet
16: Insert flow entry H_f in H
17: $H_f.p_d^* \leftarrow addr(p_d)$ ▷pointer to packet descriptor
18: $p_d.H_f \leftarrow H_f$ ▷key in hash table entry
19: Enqueue p_d at tail of Q
20: **end if**

Periodic timeout check (every P_{DT}):
21: $depth \leftarrow 0$
22: $p_d^* \leftarrow$ head of Q ▷oldest descriptor
23: $now \leftarrow$ `currentTime()`
24: **while** $p_d^* \neq$ `null` **and** $depth < d_{\max}$ **do**
25: **if** $now - p_d^*.tArr >$ DT timeout **then**
26: Delete entry H_f from hash table H
27: Forward buffered packet p to collector
28: Dequeue and free p_d^*
29: $p_d^* \leftarrow$ next descriptor in Q
30: $depth \leftarrow depth + 1$
31: **else**
32: **break** ▷subsequent entries not yet expired
33: **end if**
34: **end while**

need to install numerous rules in a short time, which can outpace the speed of MAT rule installation over the slow-path channel and the MAT memory capacity. A BF [31] is a space-efficient probabilistic data structure that tracks membership in unbounded streams with constant-time query complexity, amenable for implementation on programmable switches. In CHAMALEONET, BFs allow to (1) quickly install rules directly in the data plane without involving the control plane, and (2) filter the bulk of flows with a small memory footprint, albeit with some false positives whose impact can be made negligible (Sec. VI-C).

Standard BFs do not support deletion or stale entry removal, which is necessary for CHAMALEONET to remove stale benign flows and detect internal hosts that go dark. To address this, we adopt a rotating BF design that maintains r identical BFs. Insertions occur only in the currently active BF, while queries span all r filters to determine whether a flow has appeared within the current time window or in any of the past $r-1$ time windows. Every T_{bloom}, CHAMALEONET rotates the active BF and clears the *oldest* one, thereby removing stale entries after at most $r \cdot T_{\text{bloom}}$. Consequently, BF benign rules have a maximum lifetime of $r \cdot T_{\text{bloom}}$. After expiration, packets associated with those rules are treated as new suspicious packets and forwarded to the FSD-NF for re-evaluation. By design, this overhead is negligible: transient flows typically complete before the rotation window expires, and persistent services are promoted to MAT rules via the slow path.

We favor rotation over alternatives, such as counting BFs [32], because the latter would require the controller to maintain per-flow state to track staleness and issue explicit deletions.

3) Promotion of Long-Lived Flows to MAT Rules: we consolidate flow rules for long-lived benign flow into the slow path based on the flow lifetime. If a flow appears in $n_{\text{longlived}}$ consecutive BF's clearing events—i.e., the benign flow lifetime is at least $n_{\text{longlived}} \cdot r \cdot T_{\text{bloom}}$—the corresponding service is identified as long-lived and the control plane migrates its flow rule from the fast path to the slow-path by installing an exact-match MAT rule. To track the number of consecutive BF rotations in which a flow f appears, the control plane increases a counter every time it receives a new suspicious packet for the same flow, after the first fast-path rule for f has been installed in the BF. If the counter reaches $n_{\text{longlived}}$, the control plane promotes the flow rule to the MAT and resets the counter.

C. Traffic Anonymizer

Privacy regulations such as GDPR [12] and CCPA [13] treat IP addresses and application-layer payloads as potentially personal information. Since CHAMALEONET processes traffic generated by users connected to the campus network, it must access only the information required for its task.[4] To this end, CHAMALEONET implements configurable anonymisation directly in P4 for scalability, supporting internal IP address obfuscation and transport-layer payload removal.

We leverage the Tofino Native Architecture [33], [34]: Packets first traverse the ingress pipeline before being buffered at the Traffic Manager and processed at the egress pipeline. Processing can occur at both stages, coordinated via metadata.

[4]Privacy policies have been validated by our Data Protection Officer (DPO).

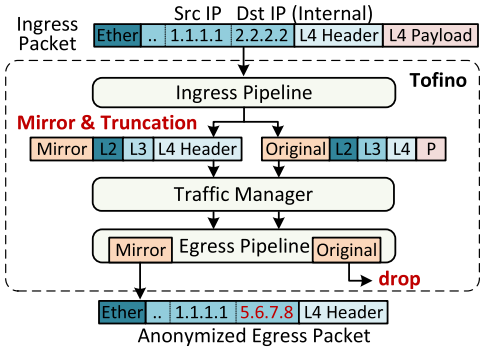


Fig. 5. P4-based traffic anonymiser implemented on an Intel Tofino [34] programmable chipset.

For IP address obfuscation of internal hosts, we rely on consistent obfuscation by XOR-ing the original value with a predetermined key salted with the last byte of the address itself. More complex techniques can be adopted [35].

While limited payload processing is feasible on programmable ASICs [36], rewriting variable-length payload remains resource-constrained. Since CHAMALEONET aims to anonymise potentially sensitive L4 payload information, we implement L4-payload anonymisation through packet truncation using mirroring sessions, as shown in Fig. 5. At ingress, the P4 program identifies packets requiring truncation, assigns them to a mirror session, and attaches internal metadata to distinguish the original packet from its mirrored copy, denoted as `Original` and `Mirror`, respectively. The controller configures the corresponding mirror session by specifying the truncation length applied at the Traffic Manager and the output port used for the mirrored packet. After truncation, both packets reach the egress pipeline, where the `Mirror` packet is forwarded to the output port connected to the FSD-NF, while the `Original` packet is dropped.

To truncate packets differently for different protocols, we define the truncation sizes either by matching the correct length in the protocol headers or by assigning pre-defined values (e.g., for unknown/unsupported protocols), each matching a different mirror session.

D. Exported Metadata and Live/Dark Host Differentiation

Unlike traditional telescopes that monitor only reserved dark subnets, CHAMALEONET collects erroneous traffic directed to both alive and dark hosts. To support downstream security analyses (Sec. V-A), CHAMALEONET exports erroneous packets (possibly anonymized/truncated according to policy) with their timestamp.

Host liveness is tracked at the SDN controller using a per-host bitmap indexed by internal IP address (live: 1, dark: 0). The controller marks a host as *alive* in the bitmap whenever it installs a long-lived MAT rule or short-lived BF rule for one of that host's flows, and it resets the bitmap every minute so that hosts that stop generating traffic transition back to *dark*. The controller exports incremental bitmap updates to the backend collector. This enables offline analytics to distinguish scans targeting dark hosts from inactive ports on live hosts.

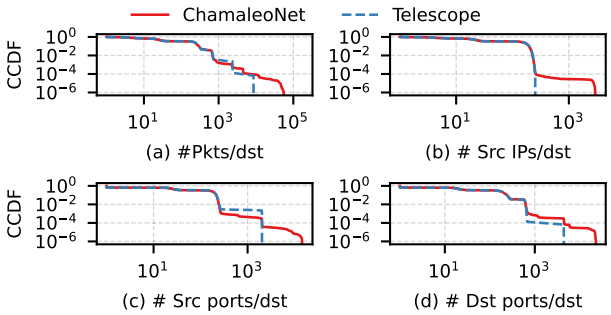


Fig. 6. Distribution of Traffic Statistics (CCDF, Log–Log Scale). (a), (b), (c), and (d) denote the number of packets, unique source IPs, unique source ports, and unique destination ports observed per destination IP address, respectively.

V. ChamaleoNet in Action

Setup. We implemented ChamaleoNet with $P4_{16}$ for the switch, and C/C++/Python for the FSD-NF and controller. We deploy ChamaleoNet on the SUP4RNET [37] testbed, on a Debian 12 server equipped with 2 Intel Xeon Gold 6252N 24-core CPUs and 128 GB of RAM, connected to the P4 switch equipped with 6.5 Tbps Wedge 100BF-32X Intel Tofino ASIC [34]. We run FSD-NF and the SDN controller on the same virtual machine (VM) featuring 16 logical cores and 16 GB vRAM. The SDN controller leverages the Barefoot Runtime Interface (BRI) to program the Tofino switch and communicates via a gRPC [38] channel. Other parameters, such as the DT timeout and BF rotation interval, are discussed in Sec. VI. According to the Tofino compiler report [33], the P4 program of ChamaleoNet consumes 26.8% SRAM, 42.0% Map RAM, and 0.7% TCAM resources.

Campus network deployment. We have deployed ChamaleoNet in a campus network for more than a year. The network is connected to the Internet with a 20 Gbps bidirectional link. At peak time, the border router forwards around 16 Gbps of total traffic. ChamaleoNet has collected data involving 34,304 internal IPv4 addresses.[5] Most of these addresses are allocated to desktops and laptops that go intermittently offline. Network Address Translation (NAT) is heavily used for both the WiFi and some department networks. Since ChamaleoNet observes traffic after NAT, outbound analysis reflects NAT-translated source IP addresses rather than individual hosts, while inbound erroneous traffic remains unaffected as scanners target publicly routable addresses. Several campus servers offer regular services on the Internet. A border firewall protects all internal hosts, allowing incoming connections only to a subset of servers and services. As such, most internal hosts and services look unreachable from remote hosts. Additionally, a /23 subnet serves as a traditional telescope. It has been operating for more than 8 years to observe only Internet radiation.

Evaluation metrics and use cases. From the collected traffic traces, we evaluate the following metrics: (i) the number of erroneous packets received; (ii) the number of unique external senders; (iii) the number of unique internal addresses contacted; (iv) the diversity of source ports used and destination ports targeted. We compare these metrics between ChamaleoNet and a /23 dark address space (pure telescope), to quantify the additional visibility gained beyond traditional telescope-based monitoring. Unless otherwise stated, we aggregate statistics over 1-hour windows. Beyond aggregate metrics, we show how these traces translate into actionable operational insights through representative incidents ChamaleoNet has observed in the wild, including amplification attacks, external routing anomalies, service misconfigurations (see Sec. V-A), and compromised internal hosts, stale DNS configurations (see Sec. V-B).

TABLE II
Quantitative Comparison Between Traffic Collected by ChamaleoNet and Pure Telescope Every Hour

Probes	Dst. IPs	Dst. Ports	Src. ASs	Src. IPs	Acknowledged Scanners
/23 telescope	510	1k	715	7k	708
ChamaleoNet	34304	62k	2310	17k	935

Traffic filtering. In our deployment, ChamaleoNet's filtering mechanism effectively suppresses over 90% of the traffic, with the FSD-NF managing just 10% of the total traffic.

A. External Erroneous Radiation

Unlike a pure telescope, ChamaleoNet monitors erroneous traffic generated by both *dark* and *live* hosts. Here, we investigate whether and how the addresses monitored by ChamaleoNet attract different traffic compared to the pure telescope. We leverage the host liveness metadata to distinguish between traffic received by live and dark hosts, as discussed in Sec. IV-D. For this study, we sample 25 days of data collected from December 7 to 31, 2024.

Sender diversity and coverage. In general, ChamaleoNet observes on average 17215 unique external senders and 1.21×10^7 erroneous flows. Table II shows the average amount of external erroneous traffic observed every hour. ChamaleoNet captures traffic from substantially more senders: $3.2\times$ more unique Autonomous System (AS), $2.4\times$ more unique IP addresses, and $1.3\times$ more acknowledged scanners.[6] This sub-linear growth is expected because many large-scale scanning campaigns probe broad portions of the IPv4 space, making most scanners already visible in the /23 deployment. Expanding coverage therefore primarily increases the interaction intensity per scanner, while newly observed scanners are mainly from small, localized probing activities (see Fig. 6 in the following). Yet, senders in ChamaleoNet are also significantly more aggressive, contacting almost the whole port range ($62\times$).

Sender aggressiveness. We analyse the distribution of traffic statistics in Fig. 6, including (a) packets, (b) distinct source IPs, (c) distinct source ports, and (d) distinct destination ports. To ensure a fair comparison between ChamaleoNet (/17) and the /23 telescope, all metrics are computed on a per-destination-IP basis.

Given the fact the most of addresses monitored by ChamaleoNet is *dark* in practice, we expect their traffic statistics to be similar to the pure telescope, as reflected by the overlapping CCDF curves. However, ChamaleoNet exhibits

[5]The campus uses one /17 and six /24 subnets, summing 34,304 addresses.

[6]Ground truth: https://gitlab.com/mcollins_at_isi/acknowledged_scanners

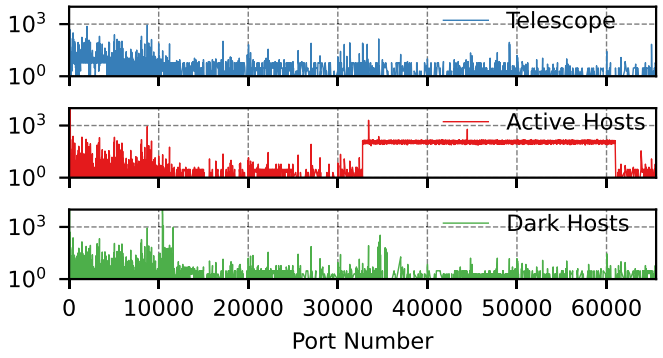


Fig. 7. Erroneous packets per destination port on the telescope, active hosts and dark hosts.

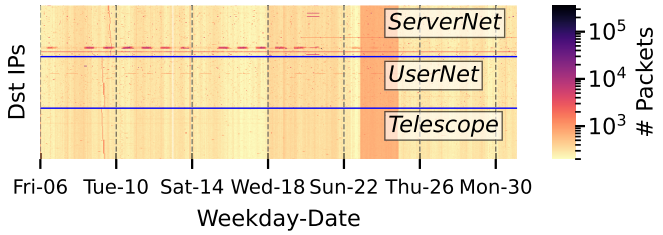


Fig. 8. Incoming traffic collected by CHAMALEONET and pure telescope (1-hour interval). Blue lines mark the three /24 subnets.

consistently heavier tails across all four metrics: the top 0.1% of destinations receive substantially more packets (up to 14×), from more distinct senders (up to 11×), with greater source-port diversity (up to 8×) and broader destination-port coverage (up to 3×). Fig. 6(b) reveals stronger concentration effects: certain hosts in CHAMALEONET are contacted by significantly more unique senders. Fig. 6(c) indicates more aggressive probing behaviours, consistent with scanners randomizing source ports or issuing repeated connection attempts. Fig. 6(d) suggests that some destinations are subjected to wider port coverage, attributing to deeper or more comprehensive scanning activity. Overall, erroneous traffic collected by CHAMALEONET exhibits consistently heavier-tailed per-destination distributions, showing increased scanner aggressiveness in terms of traffic volume and probing breadth.

Fig. 7 shows the number of packets per port observed in the top three Telescope receivers (top), active receivers (middle), and dark receivers (bottom) within CHAMALEONET over one day. The figure clearly shows that external senders target different port ranges with different intensities. While well-known ports and low port numbers receive more radiation, the presence of active services causes a significant increase in unsolicited traffic, causing the per-port probability to change.

In a nutshell, CHAMALEONET collects much more Internet radiation than a pure telescope. As [7], [8] transformed a CDN and a cloud provider into a telescope, CHAMALEONET transforms any live network into a better telescope, without the need to reserve precious IP addresses for this.

Temporal evolution. We now examine the temporal evolution of external erroneous radiation using two representative /24 subnets monitored by CHAMALEONET. *ServerNet* is a /24 subnet hosting 10-15 campus servers, the other addresses acting as dark hosts. *UserNet* is a /24 subnet with client hosts that are unreachable from outside. For comparison, we also include one /24 telescope subnet. Fig. 8 illustrates the erroneous packets for each address in *ServerNet*, *UserNet*, and *Telescope*, separated by solid blue lines.

Scan pattern is very similar, with different intensities–notice the 48-hour intensive scanning event during Dec. 23–24 period. Yet, the *ServerNet* shows clearly some different pattern: the continuous horizontal line refers to active servers that external actors keep scanning on several ports. Some intermittent pattern related to human activity emerges. Here, thousands of external addresses send erroneous packets to the public addresses allocated to the campus WiFi network NAT. These packets are due to ongoing connections where the internal WiFi host suddenly becomes unreachable, and the external senders keep retransmitting packets. Additionally, in the *ServerNet* we observe some sudden and prolonged increase in erroneous traffic (small red horizontal bars). These are external scanners that look for other servers in this /24 subnet.

External senders targeting campus addresses. During the study period, we observed a total of 436k unique external senders. Among them, 56.7% target campus addresses exclusively and avoid any telescope address (while only 0.3% interact solely with the telescope). By analysing the top-senders, we identify a variety of behaviours, including attack patterns, scanning activities, misconfigurations, etc. We describe some notable findings below (with anonymized addresses).

- IP 5.96.X.X: This is the most active sender. It generates an average of 3,000 packets per hour for the entire period. All packets go to UDP port 123 on a specific internal server, strongly suggesting a Network Time Protocol (NTP) attack.
- IP 188.92.X.X: It scans ports 37215, 52869, and 49152 on thousands of hosts, hinting at a Satori bot [39]. It also scans other ports (e.g., 1900 and 2048) indicating broader scanning. The sender is present in blocklists.
- IP 193.X.X.X: It belongs to the campus network provider. It keeps sending more than 1,000 ICMP Time Exceeded messages per hour, targeting 336 internal addresses. This suggests a routing issue (or blackholed destination). Internal clients cannot reach the final destination.
- IP 94.143.X.X: It consistently contacts a port on an internal server from source port 57498, sending about 720 packets per hour. This indicates a misconfigured or failed service.
- 38 IP addresses (e.g., 49.232.X.X) conduct synchronised ICMP scans throughout the period, sending an average of 394 ICMP Echo Requests per hour to 20 internal IP addresses. The campus firewall blocks ICMP echo requests.

Firing impersonators to engage with scanners. To understand which services the scanners are interested in, we activate TCP impersonator on some unused IP addresses on the campus for three hours. These impersonators complete the TCP three-way handshake, observe the first application message, if any, and tear down the connection. For impersonation traffic, CHAMALEONET operates in a pass-through mode with explicit redirection. The switch bypasses filtering and anonymization and forwards traffic to FSD-NF directly. In this

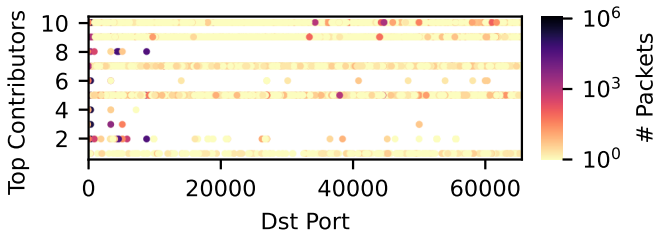


Fig. 9. TCP destination ports of internal erroneous traffic generated by the top-10 contributors.

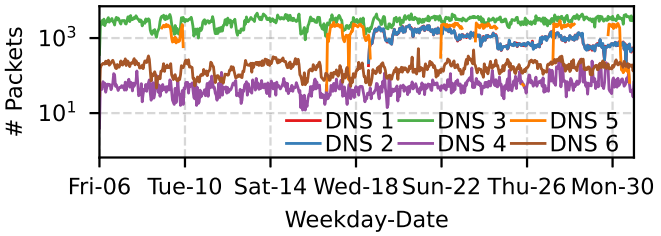


Fig. 10. Erroneous DNS traffic sent by the campus DNS servers to external DNS servers.

mode, FSD-NF disables buffering and detection logic and forwards traffic to the responders, while simultaneously mirroring a copy to the collector for deeper analysis. We run nDPI [2] to extract the application protocol classification of captured traffic, focusing on traffic to ports in the $[30000, 61000]$ range. Results show a mix of different protocols, the majority unknown to nDPI (45%) (because only the first payload is captured), or has no payload (21%) (because of client-initiated protocol), with HTTP (18%), TLS (4%), LDAP, DRDA, DRP, SQL, PRTP and other protocols being probed (<1%). This confirms that the external scanners probe for various services on non-standard ports once they find the host alive [20]. The impersonating ability of CHAMALEONET is fundamental for exploring the intention of senders.

B. Internal Erroneous Radiation

Let us focus on the erroneous packets generated by internal hosts that CHAMALEONET captures. Recall that a pure telescope will not collect any information on this. On average, we observe approximately 40 k outgoing erroneous packets per hour, consisting of roughly 40% TCP packets, 36% UDP packets, and 24% ICMP packets. Again, we use simple analytics to quickly highlight suspicious behaviours.

Top senders of internal erroneous traffic. We look at those hosts that contribute the most to the erroneous radiation. We consider one day of traffic. Each of the top senders contributes tens of thousands of erroneous packets. Fig. 9 shows the destination ports targeted by the top-10 senders, the most active on the bottom. Two patterns emerge: (i) horizontal scan patterns that target all (e.g., 1st, 4th-6th senders) or most ports (e.g., 9th sender); (ii) vertical scan patterns, targeting few ports (e.g., 3rd, 7th senders). Some vertical scanners constantly send thousands of unanswered requests per hour to very few IP addresses in a cloud system. This hints at misconfigured systems that keep sending traffic to non-existent servers (some examples later). For horizontal scanners, some target a few external hosts while others contact hundreds of different destinations. This hints at bots or malicious scanning activities that we signalled to our IT security team.

Erroneous DNS traffic. By looking at the most targeted ports, we notice a lot of erroneous UDP traffic destined to port 53/DNS and sent by the campus official DNS resolvers. Digging into this, we observe that most of this traffic is directed to a few authoritative DNS resolvers (in Iran, China, India), which never responded to our resolvers. We report the temporal evolution of this erroneous traffic in Fig. 10. We suspect this suspicious activity is part of attacks where infected internal clients send DNS requests to our resolvers that route them to the authoritative resolvers. Again, we signalled the incident to our IT security team.

Other findings. Among the recurrent patterns exposed by CHAMALEONET, we observe occasional peaks of unanswered traffic to popular external servers. Unlike the previous cases, these patterns stem from benign communications but exhibit erroneous behaviour, likely due to misconfigurations or transient operational issues. Since CHAMALEONET targets operational visibility rather than attack detection, we treat these events as visibility signals of abnormal network conditions. For example, we observe a steady stream of TCP SYN packets, about 3.7k per hour, from a specific internal server to a remote HPC repository. Investigation revealed that the repository had been decommissioned, while its DNS name remained registered, causing the server to keep reconnecting.

Another case involves over 50 internal hosts generating unusual ICMP port unreachable messages. We traced them to DNS acceleration tools that send parallel queries to multiple open resolvers and close the socket after the first response. Later responses then arrive at closed sockets, causing ICMP port unreachable messages that CHAMALEONET correctly logs as erroneous traffic.

Understanding the precise reasons for these behaviours is outside our scope. However, these simple findings illustrate the benefits of the visibility CHAMALEONET provides.

VI. SYSTEM PERFORMANCE EVALUATION

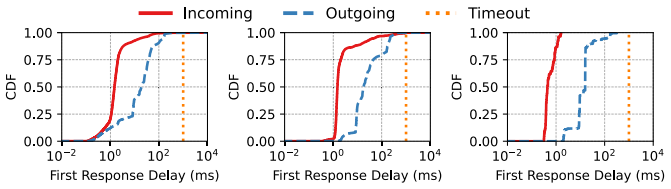
We now present results collected in controlled experiments to optimize CHAMALEONET design, select system parameters and characterize scalability.

A. Parameter Setting

We set up a second VM to run a traffic generator, bridged to the server E810 NIC in PCIe passthrough mode using a separate SR-IOV [40] virtual function. Packets transmitted from the traffic generator VM are forwarded to the P4 switch before returning to the FSD-NF VM.

Replay traffic workload (TW-R). We run a traffic replay workload that uses realistic traffic patterns captured from a campus network, containing about 6 million flows and 152 million packets over 10 minutes. We use up to 10 `tcpreplay` [41] instances to generate different traffic rates, splitting and multiplexing traces in parallel to mimic real traffic patterns. We label this workload as TW-R for later reference.

DeTection (DT) timeout. For setting the DT timeout we observe the empirical response delay statistics of actual flows.



(a) TCP (b) UDP (c) ICMP

Fig. 11. First response delay of incoming and outgoing flows, measured in our campus network.

We analyse the campus traces to measure the *first* response delay of answered flow, i.e., the time elapsed between the first packet and the first response packet. We observe more than 100,000, 72,000 and 1,000 TCP, UDP and ICMP flows, respectively. About one-tenth are incoming, i.e., requests launched by external clients targeting internal services.

Fig. 11 shows the Cumulative Distribution Function (CDF) of the response delay for TCP, UDP, and ICMP, respectively. The CDFs show that about 80% of incoming requests are answered in a few ms. This is expected since internal hosts located within the main campus LAN have a short Round Trip Time (RTT). Because our campus also includes some remote laboratories connected with Virtual Private Networks (VPNs) over the Internet, some clients and servers suffer longer RTT. Looking instead at outgoing flows, the response time is significantly higher as it includes the RTT to servers located anywhere on the Internet.

The choice of a proper DT timeout is a balance between accuracy and system load: increasing the timeout improves flow identification accuracy, but impacts system load and scalability. In our setting, we set the timeout to 1s to correctly classify over 99.9% incoming and 99.8% outgoing answered flows. Note that ChamaleoNet could be easily extended to support multiple timeouts by simply splitting traffic to separate NFs or packet descriptor queues, e.g., using different DTs for incoming and outgoing requests, as done in telescope-like systems [8], [19], [42].

DT timeout expiration routine: tuning (P_{DT}, $d_{\max}$). As elaborated in Sec. IV, pkt_rx handles both incoming suspicious packets and DT timeout expiration. Every P_{DT} seconds, packet reception is briefly interrupted to check the HoL descriptor and pop expired entries from the ring queue, up to a maximum scanning depth $d_{\max}$. Longer checks increase the time packets spend waiting in the `libpcap` buffer, although this does not affect correctness as long as no packet is dropped.

We evaluate the impact of $(P_{DT}, d_{\max})$ on per-packet processing time and DT buffering accuracy, shown in the top and bottom subfigures of Fig. 12, respectively. We sample one out of every $10,000$ packets and, under $10\times$ `tcpreplay` sender load, configure all settings to perform timeout expiration checks at the same aggregate rate, i.e., $d_{\max}/P_{DT} = 1$ M checks/s. Fig. 12 reports the 75th and 99th percentiles across different configurations.

Frequent, fine-grained checks provide the best tradeoff. With $P_{DT} = 0.01$ ms and $d_{\max} = 10$, 99% of packets are processed within 3 ms with no packet loss, compared to 55 ms when checking up to 10,000 descriptors every 10 ms. The

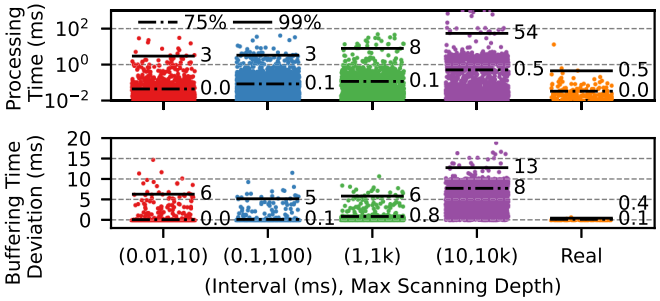


Fig. 12. Periodic check of timeout expiration for various configurations of $(P_{DT}, d_{\max})$, with DT timeout set to $1s$. "Real" refers to the deployment in our campus network.

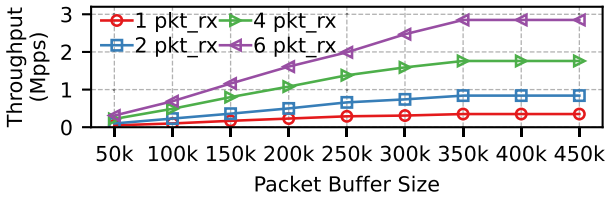


Fig. 13. Throughput of the FSD-NF under DoS workload.

buffering-time results show negligible deviation from the predefined DT timeout: for example, $P_{DT} = 0.1$ ms and $d_{\max} = 100$ keeps 99% of packets buffered for about 1.005 s. Results with real traffic traces confirm the same trend.

These results reveal a broad stable operating region for $(P_{DT}, d_{\max})$. We therefore adopt fixed parameters selected empirically from this region.

B. Scalability Analysis

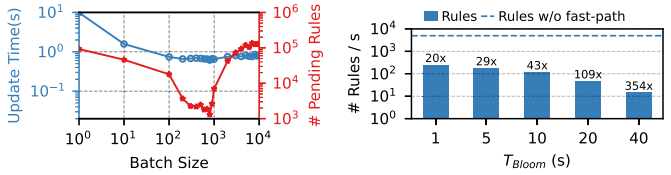
In this section, we evaluate ChamaleoNet's performance under adversarial conditions that stress both the packet processing and rule management logic.

Adversarial traffic workloads. We create two synthetic adversarial traffic workloads generated with Cisco T-Rex [43], a state-of-the-art traffic generator in DPDK [30]:

- *DoS traffic* (TW-DoS): every packet corresponds to a new flow with no response. This workload stresses FSD-NF's packet processing throughput by maximizing the number of concurrent erroneous flows.
- *High-churn transient benign traffic* (TW-HCB): every flow consists of exactly two bidirectional packets, a legitimate request and its corresponding response. This workload stresses both the FSD-NF's packet processing throughput and the rule management logic on the fast-path, by rapidly generating new, short-lived benign flows.

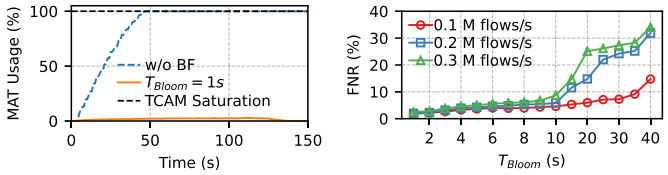
For both workloads, we generate traffic with minimum packet size to maximize the packet processing load on FSD-NF (i.e., 64 bytes on Ethernet).

FSD-NF packet processing throughput. The first threat to scalability is FSD-NF's packet processing throughput. When all incoming packets correspond to new flows, FSD-NF receives large bursts of packets, and high memory pressure to store flow states. We benchmark FSD-NF's peak throughput following RFC 2544 [44], stopping at the first packet loss. Fig. 13 shows the peak throughput achieved with 1, 2, 4, and 6 pkt_rx threads, under different sizes of the ring buffer



(a) (b)

Fig. 14. (a) Batching amortizes gRPC call overhead, until the switch's processing time for rule installation becomes dominant. (b) The fast-path cuts the update rate on the slow-path by orders of magnitude, mitigating bottleneck *#1*.



(a) (b)

Fig. 15. (a) The Bloom filter (BF) bounds TCAM occupancy, whereas the baseline saturates the capacity shortly. (b) Empirical false negative rate of CHAMALEONET's filtering across different T_{bloom} under adversarial traffic.

in the FSD-NF. As the DoS and high-churn transient benign workloads show similar results, we omit the latter for brevity. For small packet buffer sizes, the throughput is limited by early buffer fill-up. For buffer size larger than ∼350k packets, the overheads for packet processing operations become dominant (e.g., buffering, expiry checking, and garbage collection). Packets cannot be drained from the ring buffer fast enough, resulting in packet drops and throughput saturation. Overall, FSD-NF reaches a peak throughput of 350, 840, 1760, and 2850 kpps with one, two, four, and six threads, respectively, consuming about 16 GB of vRAM with six threads. This is 519× the erroneous traffic we normally receive in our deployment at the NF. These results demonstrate linear scaling of CHAMALEONET's throughput when packet processing is parallelized across six pkt_rx threads.

Filtering rule management. The second threat to CHAMALEONET's scalability is represented by the rule management mechanism. Here, scalability is constrained by two bottlenecks: *(i)* the control-plane channel bandwidth for rule updates and *(ii)* the switch's memory capacity to store flow rules. We analyse each of these bottlenecks in turn, highlight when they emerge, and show that CHAMALEONET's design effectively mitigates both.

Bottleneck #1: gRPC channel for slow-path rule updates. Under traffic with a prevalence of long-lived flows, the gRPC control-plane channel may become the bottleneck due to the high rate of rule updates. CHAMALEONET's design includes a fast-path filtering mechanism to mitigate this issue. We quantify its effectiveness as follows.

First, we run a controlled micro-benchmark to measure the maximum throughput for rule update on the slow path (i.e., gRPC channel). We run a control-plane script that continuously installs rules on the Tofino ASIC via the BFrt [45] control-plane APIs. To maximize throughput, we batch multiple rule updates in the same gRPC request. We empirically tune the batch size to a value that amortizes the cost of the gRPC calls (maximizing throughput), without incurring excessive latency from bulk rule installation at the switch. Based on the measurements shown in Fig. 14a, we set the batch size to 200 rules, a value chosen in the sweet spot region. With this value, we measure a maximum throughput of around 14k rule updates/sec, which we take as a reference capacity of the gRPC channel for the following experiment.

Second, we run CHAMALEONET *with* and *without* the fast-path enabled. We generate a traffic workload that stresses the gRPC control-plane channel by generating a large number of long-lived flows, which require slow-path rule updates. To do so, we replay traffic TW-HCB and scale the number of replicas until we saturate the gRPC channel capacity, measured in the earlier experiment. When the fast-path is enabled, we set $n_{\text{longlived}} = 2$ for the long-lived flow promotion (discussed in Sec. IV-B3). Fig. 14b shows that the fast-path can mitigate the slow path bottleneck, by cutting the rule update rate by up to 354×. Longer BF rotation intervals T_{bloom} lead to lower slow-path update rates, as more transient flows are filtered in the fast path and do not require promotion to the slow path.

Bottleneck #2: Switch memory usage. The second bottleneck is the switch's memory capacity to store flow rules. Hence, we quantify the impact of the fast-path scheme on the TCAM usage in Fig. 15a. The plot shows the number of rules installed in the switch's TCAM over time, with and without the fast-path enabled. With the fast-path enabled, the number of installed rules remains stable and low, peaking below 25k MAT entries (2.8% of the 80k allocated entries), compared to 99.96% without the fast-path. This is because short-lasting flows that do not persist across multiple BF rotation epochs are not promoted, while only stable services are installed as MAT entries. We next examine whether moving transient benign-flow rules to the fast path suffers from a memory bottleneck.

C. Impact of Bloom Filter (BF) on Visibility

Storing benign flow rules installed via the fast-path on Bloom filters consumes SRAM resources. Because SRAM memory is scarce in programmable switches (a few dozens of MB on Tofino), here we evaluate the trade-off between memory usage and visibility of erroneous traffic.

Increasing T_{bloom} keeps benign-flow rules in the BFs for longer and therefore increases BF occupancy. BF occupancy determines collision probability in the BF and drives the visibility trade-off: a false positive in any of the r BFs can incorrectly match an erroneous packet to a benign flow entry, causing the packet to be discarded, i.e., a false negative for CHAMALEONET.[7]

Empirical False Negative Rate (FNR). Fig. 15 (b) evaluates the visibility degradation of CHAMALEONET under adversarial traffic by reporting the empirical false negative rate. We construct a traffic workload that mixes the replay traffic workload TW-R with different proportions of high-churn transient benign traffic TW-HCB. We then measure an

[7]The false negative rate (FNR) in CHAMALEONET can be computed as $\text{FNR} = 1 - (1 - \text{FPR}_{BF})^r$, where $\text{FPR}_{BF} = (1 - e^{-kn/m})^k$ is the standard BF false positive rate for k hash functions, n elements, and m-bit array.

empirical FNR that quantifies the amount of "lost" erroneous traffic, *i.e.*, erroneous traffic mistakenly dropped by the BFs. To measure the empirical FNR, we first identify the ground-truth erroneous traffic in the workload we generate, then we compare it with the erroneous traffic logged by ChamaleoNet. We use r=2 BFs, each with 2 hash functions and a bit array of 2^{22} entries. We set these parameters to the largest values allowed by the compiler and per-stage resource constraints of Tofino. As shown in Fig. 15 (b), the FNR grows with T_{bloom} and the proportion of high-churn benign traffic, confirming the expected trade-off. At a churn rate of 0.3 M flows/s, *i.e.*, around the saturation point of a single pkt_rx thread, setting $T_{\text{bloom}} = 1s$ leads to FNR 2.4%. In practice, the actual flow churn is over an order of magnitude lower on average in a campus network (see operating point below). In addition, we observe a region, for $T_{\text{bloom}} \leq 10s$, where the empirical FNR grows slowly, and remains at acceptable levels. This suggests that ChamaleoNet can be configured to operate in a regime where the BF provides effective filtering while keeping visibility degradation under control.

Operating point (campus network). *In our real deployment, the benign-flow churn rate is around 11.8 k flows/s on average. We set $T_{\text{bloom}} = 5s$, which corresponds to an empirical FNR of 5.0% under the adversarial workload. With this setting, the rule update rate on the slow-path is reduced by 36.1%, resulting into 2454 rules/sec on average. Correspondingly, the load on the FSD-NF is reduced by 90.0%, resulting into 76.4 kpps received traffic, which can be handled by one* pkt_rx *thread.*

VII. Discussion

Larger deployments. We evaluated a single switch deployment, and demonstrated that our current design suffices for a campus-network size. To scale to larger deployments, such as ISP or data center size, network managers could shard ChamaleoNet's dataplane across multiple P4 switch replicas, to further scale out SRAM memory for BFs, and gRPC channel capacity for long-rule installation. ChamaleoNet's design is compatible with sharding. Each switch replica can run the same P4 program, and flows can be distributed across replicas using standard affinity-preserving per-flow load balancing. The control-plane can also be replicated to manage each switch independently. Lastly, and orthogonally to this paper, systems optimizations [46] for high-speed stateful network functions can be adopted to further improve scalability of FSD-NF under multi-threading, by mitigating memory access contentions that inevitably arise as thread count increases. We regard these as implementation challenges rather than a fundamental scalability limit of ChamaleoNet's design.

Offloading FSD-NF to programmable dataplanes. Moving FSD-NF entirely into the data plane could eliminate the software FSD-NF bottleneck entirely. However, this design faces two fundamental challenges. First, simultaneously buffering suspicious packets of multiple flows for *tens of milliseconds* is infeasible at modern link speeds, given the limited on-chip memories. For example, a 10MB buffer, compatible with a typical data center switch memory size [47], [48], would fill in about 1ms at 100Gbps. Second, programmable switch ASICs lack random-access memory, making it non-trivial to retrieve a buffered suspicious packet once it should be discarded. Integrating ChamaleoNet with state-of-the-art work [36] is left for future enhancement.

In-path deployment. ChamaleoNet's architecture relies on mirroring ingress/egress traffic from the campus network to the SDN switch, resulting in an out-of-path deployment. Alternatively, ChamaleoNet could be deployed in-path on SDN-capable border router (or any downstream switch on the ingress/egress path). In such cases, the FSD-NF shall observe all suspicious packets and install a rule to *forward* benign flow packets to the campus network, instead of *dropping* them.

In-path deployment enables the defense from erroneous flows, *i.e.*, dropping or re-routing erroneous packets instead of forwarding them to the campus network. However, in-path deployment introduces extra challenges, such as invertible anonymisation and additional forwarding latency on regular traffic. It is also more vulnerable to false negatives: missing the detection of a benign flow would disrupt regular traffic, since no forwarding rule would be installed. Consequently, network administrators may be reluctant to introduce ChamaleoNet on the live traffic path. The out-of-path design illustrated in Fig. 2 offers a less intrusive and pluggable solution.

VIII. Ethical Considerations

Privacy. Privacy is fundamental as legitimate traffic is also mirrored. We consider an "honest but curious" scenario where the person in charge of processing the data will not deviate from the defined goals. We thus adopt a *data minimisation approach*: To minimise the information the controller and the collectors process and store, we keep only headers up to the transport protocol and remove any upper-layer information. We support IP address anonymisation for the internal network IP addresses by using a simple obfuscation mechanism. We leverage the data-plane programmability features to perform anonymisation directly at the switch (see Sec. IV-C).

Impersonating not-responding servers or services. ChamaleoNet cannot interfere with production traffic. Logging erroneous packets is not critical–impersonating non-responding servers or services requires particular attention. When internal services go offline for maintenance or temporary failures, legitimate users could find themselves interacting with honeypots. Likewise, impersonating an internal offline client (e.g., a PC turned off at night) poses legal and security concerns if its address is converted into honeypot. We limit ourselves to demonstrating the ChamaleoNet impersonating ability with some pre-determined off-line addresses, leaving its extensive usage for future analysis. Notice that the privacy features must be disabled for the hosts/services ChamaleoNet will impersonate so that the original IP addresses and payload are exposed to the impersonator.

IX. Conclusion

We presented ChamaleoNet, a system that transforms any production network into a programmable probe to monitor erroneous traffic–such as unanswered, misrouted, or malformed packets. By leveraging SDN principles and programmable data planes, ChamaleoNet offers scalable, privacy-compliant traffic visibility without disrupting normal

network operations. It operates transparently in live environments, collecting only erroneous traffic, anonymising sensitive information at the switch, and enabling both passive observation and active engagement through honeypot integration.

Our deployment over a year showed CHAMALEONET's effectiveness in uncovering external scanning behaviours and internal misconfigurations, outperforming traditional telescope setups in data richness and insights. It filtered up to 90% of benign traffic at the hardware level, ensuring scalability with minimal resource requirements. CHAMALEONET is open source, modular, and ready for deployment in various environments, offering a practical and ethical solution to enhance network visibility and threat intelligence without requiring dedicated addresses or additional instrumentation.

DISCLAIMER

This manuscript reflects only the authors' views and opinions and the Ministry cannot be considered responsible for them.

REFERENCES

[1] M. Trevisan, A. Finamore, M. Mellia, M. Munafò, and D. Rossi, "DPDKStat: 40 Gbps statistical traffic analysis with off-the-shelf hardware," Telecom ParisTech, Paris, France, Tech. Rep. 318627, 2016. [Online]. Available: https://nonsns.github.io/paper/DPDKStat-techrep.pdf

[2] L. Deri, M. Martinelli, T. Bujlow, and A. Cardigliano, "NDPI: Open-source high-speed deep packet inspection," in *Proc. Int. Wireless Commun. Mobile Comput. Conf. (IWCMC)*, Aug. 2014, pp. 617–622.

[3] T. Zhang, L. Linguaglossa, M. Gallo, P. Giaccone, and D. Rossi, "FloWatcher-DPDK: Lightweight line-rate flow-level monitoring in software," *IEEE Trans. Netw. Service Manage.*, vol. 16, no. 3, pp. 1143–1156, Sep. 2019.

[4] O. H. Abdulganiyu, T. A. Tchakoucht, and Y. K. Saheed, "A systematic literature review for network intrusion detection system (IDS)," *Int. J. Inf. Secur.*, vol. 22, no. 5, pp. 1125–1162, Oct. 2023.

[5] R. Pang, V. Yegneswaran, P. Barford, V. Paxson, and L. Peterson, "Characteristics of internet background radiation," in *Proc. 4th ACM SIGCOMM Conf. Internet Meas.*, Oct. 2004, pp. 27–40.

[6] D. Moore, C. Shannon, G. M. Voelker, and S. Savage, "Network telescopes," Dept. Comput. Sci. Eng., Univ. California, San Diego, CA, USA, Tech. Rep. CS2004-0795, 2004.

[7] E. Pauley, P. Barford, and P. McDaniel, "DScope: A cloud-native internet telescope," in *Proc. 32nd USENIX Secur. Symp. (USENIX Secur. 23)*, Aug. 2023, pp. 5989–6006. [Online]. Available: https://www.usenix.org/conference/usenixsecurity23/presentation/pauley

[8] P. Richter and A. Berger, "Scanning the scanners: Sensing the internet from a massively distributed network telescope," in *Proc. Internet Meas. Conf.*, Oct. 2019, pp. 144–157. [Online]. Available: https://dl.acm.org/doi/10.1145/3355369.3355595

[9] D. Wagner et al., "How to operate a meta-telescope in your spare time," in *Proc. ACM IMC*. New York, NY, USA: Association for Computing Machinery, 2023, pp. 328–343, doi: 10.1145/3618257.3624831.

[10] S. Guo et al., "Skyline: A cloud centric internet monitoring engine," in *Proc. USENIX NSDI*, May 2026, pp. 685–699. [Online]. Available: https://www.usenix.org/conference/nsdi26/presentation/guo-shixian

[11] A. Sordello, Z. Wang, K. Huang, A. Cornacchia, and M. Mellia, "Poster: The potential of erroneous outbound traffic analysis to unveil silent internal anomalies," in *Proc. ACM Internet Meas. Conf.* New York, NY, USA: Association for Computing Machinery, Oct. 2025, pp. 1078–1079, doi: 10.1145/3730567.3768599.

[12] Eur. Parliament Council Eur. Union. (2016). *Regulation (EU) 2016/679: General Data Protection Regulation*. Accessed: Aug. 10, 2025. [Online]. Available: https://eur-lex.europa.eu/eli/reg/2016/679/oj

[13] California State Legislature. (2018). *California Consumer Privacy Act of 2018*. Accessed: Aug. 10, 2025. [Online]. Available: https://leginfo.legislature.ca.gov/faces/billTextClient.xhtml?bill/id=201720180AB375

[14] D. Ergenç, R. Schenderlein, and M. Fischer, "TSNZeek: An open-source intrusion detection system for IEEE 802.1 time-sensitive networking," in *Proc. IFIP Netw. Conf. (IFIP Netw.)*, Jun. 2023, pp. 1–6.

[15] The Zeek Project. (2025). *Zeek Documentation: Newconnection Event*. Accessed: Aug. 10, 2025. [Online]. Available: https://docs.zeek.org/en/current/scripts/base/bif/event.bif.zeek.html#id-newconnection

[16] M. Jonker, A. King, J. Krupp, C. Rossow, A. Sperotto, and A. Dainotti, "Millions of targets under attack: A macroscopic characterization of the DoS ecosystem," in *Proc. Internet Meas. Conf.*, Nov. 2017, pp. 100–113. [Online]. Available: http://dl.acm.org/citation.cfm?doid=3131365.3131383

[17] M. Antonakakis et al., "Understanding the Mirai botnet," in *Proc. 26th USENIX Secur. Symp. (USENIX Security)*, 2017, pp. 1093–1110. [Online]. Available: https://www.usenix.org/conference/usenixsecurity17/technical-sessions/presentation/antonakakis

[18] Center for Applied Internet Data Analysis (CAIDA). (2025). *The UCSD Network Telescope*. Accessed: Aug. 10, 2025. [Online]. Available: https://www.caida.org/projects/networktelescope/

[19] R. Hiesgen, M. Nawrocki, A. King, A. Dainotti, C. T. Schmidt, and M. Wählisch, "Spoki: Unveiling a new wave of scanners through a reactive network telescope," in *Proc. 31st USENIX Secur. Symp.*, 2022, pp. 431–448. [Online]. Available: https://www.usenix.org/conference/usenixsecurity22/presentation/hiesgen

[20] F. Soro et al., "Enlightening the darknets: Augmenting darknet visibility with active probes," *IEEE Trans. Netw. Service Manage.*, vol. 20, no. 4, pp. 5012–5025, Dec. 2023.

[21] P. Hadem, D. K. Saikia, and S. Moulik, "An SDN-based intrusion detection system using SVM with selective logging for IP traceback," *Comput. Netw.*, vol. 191, May 2021, Art. no. 108015. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1389128621001274

[22] B. Ayodele and V. Buttigieg, "SDN as a defence mechanism: A comprehensive survey," *Int. J. Inf. Secur.*, vol. 23, no. 1, pp. 141–185, Feb. 2024, doi: 10.1007/s10207-023-00764-1.

[23] R. Li, M. Zheng, D. Bai, and Z. Chen, "SDN based intelligent honeynet network model design and verification," in *Proc. Int. Conf. Mach. Learn. Intell. Syst. Eng. (MLISE)*, Jul. 2021, pp. 59–64.

[24] M. Karakate, H. Esaki, and H. Ochiai, "SDNHive: A proof-of-concept SDN and honeypot system for defending against internal threats," in *Proc. ACM ICCNS*, 2022, pp. 9–20.

[25] P. Bosshart et al., "P4: Programming protocol-independent packet processors," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 3, pp. 87–95, Jul. 2014, doi: 10.1145/2656877.2656890.

[26] A. Bianco, P. Giaccone, S. Kelki, N. M. Campos, S. Traverso, and T. Zhang, "On-the-fly traffic classification and control with a stateful SDN approach," in *Proc. IEEE Int. Conf. Commun. (ICC)*, May 2017, pp. 1–6.

[27] F. Paolucci, F. Civerchia, A. Sgambelluri, A. Giorgetti, F. Cugini, and P. Castoldi, "P4 edge node enabling stateful traffic engineering and cyber security," *J. Opt. Commun. Netw.*, vol. 11, no. 1, pp. A84–A95, Jan. 2019. [Online]. Available: https://opg.optica.org/jocn/abstract.cfm?URI=jocn-11-1-A84

[28] M. Monshizadeh, V. Khatri, and R. Kantola, "Detection as a service: An SDN application," in *Proc. 19th Int. Conf. Adv. Commun. Technol. (ICACT)*, Feb. 2017, pp. 285–290.

[29] P. Ferguson and D. Senie, *Network Ingress Filtering: Defeating Denial of Service Attacks Which Employ IP Source Address Spoofing*, document RFC 2827, May 2000. [Online]. Available: https://www.rfc-editor.org/info/rfc2827

[30] Intel Corporation. (2025). *Data Plane Development Kit (DPDK)*. Accessed: Aug. 10, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/developer/topic-technology/networking/dpdk.html

[31] W. Wang, X. C. Wu, P. Tammana, A. Chen, and T. S. E. Ng, "Closed-loop network performance monitoring and diagnosis with SpiderMon," in *Proc. 19th USENIX Symp. Networked Syst. Design Implement. (NSDI 22)*, Apr. 2022, pp. 267–285. [Online]. Available: https://www.usenix.org/conference/nsdi22/presentation/wang-weitao-spidermon

[32] J. Hill, M. Aloserij, and P. Grosso, "Tracking network flows with P4," in *Proc. IEEE/ACM Innovating Netw. Data-Intensive Sci. (INDIS)*, Nov. 2018, pp. 23–32.

[33] Intel Corporation. (2021). *P416 Intel Tofino Native Architecture—Public Version*. Accessed: Aug. 10, 2025. [Online]. Available: https://github.com/barefootnetworks/Open-Tofino/blob/master/PUBLICTofino-Native-Arch.pdf

[34] (2025). *Intel Tofino Series*. Accessed: Aug. 10, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/products/network-io/programmable-ethernet-switch.html

[35] X. Chen, "Implementing AES encryption on programmable switches via scrambled lookup tables," in *Proc. Workshop Secure Program. Netw. Infrastruct.*, Aug. 2020, pp. 8–14, doi: 10.1145/3405669.3405819.
[36] S. Goswami, N. Kodirov, C. Mustard, I. Beschastnikh, and M. Seltzer, "Parking packet payload with P4," in *Proc. 16th Int. Conf. Emerg. Netw. Exp. Technol.* New York, NY, USA: Association for Computing Machinery, Nov. 2020, pp. 274–281, doi: 10.1145/3386367.3431295.
[37] PROGNOSE Lab. (2025). *SUP4RNET: An Experimental Platform Running on P4 Switches Inside the PROGNOSE Lab*. Accessed: Aug. 10, 2025. [Online]. Available: https://sup4rnet.github.io
[38] gRPC Authors. (2025). *GRPC*. Accessed: Aug. 10, 2025. [Online]. Available: https://grpc.io
[39] A. A. Al Alsadi et al., "No spring chicken: Quantifying the lifespan of exploits in IoT malware using static and dynamic analysis," in *Proc. ACM Asia Conf. Comput. Commun. Secur.* New York, NY, USA: Association for Computing Machinery, May 2022, pp. 309–321, doi: 10.1145/3488932.3517408.
[40] Intel Corporation. (2025). *SDN-NFV Hands-on Samples*. Accessed: Aug. 10, 2025. [Online]. Available: https://github.com/intel/SDN-NFV-Hands-on-Samples
[41] (2025). *Tcpreplay: Pcap Editing and Replay Tools for Network Testing*. Accessed: Aug. 10, 2025. [Online]. Available: https://tcpreplay.appneta.com/
[42] H. Griffioen, G. Koursiounis, G. Smaragdakis, and C. Doerr, "Have you SYN me? Characterizing ten years of internet scanning," in *Proc. ACM Internet Meas. Conf.* New York, NY, USA: Association for Computing Machinery, Nov. 2024, pp. 149–164, doi: 10.1145/3646547.3688409.
[43] Cisco Systems.(2025). *TRex: Realistic Traffic Generator*. Accessed: Aug. 10, 2025. [Online]. Available: https://trex-tgn.cisco.com/
[44] S. Bradner and J. McQuaid, *Benchmarking Methodology for Network Interconnect Devices*, document RFC 2544, Mar. 1999.
[45] Intel Corporation. (2025). *SDE 9.13.0 BF-Runtime GRPC Doxygen*. Accessed: Aug. 10, 2025. [Online]. Available: https://www.intel.com/content/www/us/en/secure/content-details/778340/sde-9-13-0-bf-runtime-grpc-doxygen.html
[46] H. Ghasemirahni, A. Farshin, M. Scazzariello, G. Q. Maguire Jr., D. Kostić, and M. Chiesa, "FAJITA: Stateful packet processing at 100 million pps," in *Proc. ACM Netw.*, vol. 2, Sep. 2024, pp. 14:1–14:22.
[47] D. Kim et al., "TEA: Enabling state-intensive network functions on programmable switches," in *Proc. Annu. Conf. ACM Special Interest Group Data Commun. Appl., Technol., Archit., Protocols Comput. Commun.* New York, NY, USA: Association for Computing Machinery, Jul. 2020, pp. 90–106, doi: 10.1145/3387514.3405855.
[48] M. Moshref, M. Yu, R. Govindan, and A. Vahdat, "SCREAM: Sketch resource allocation for software-defined measurement," in *Proc. ACM CoNEXT*, 2015, pp. 14:1–14:13.



Zhihao Wang received the Ph.D. degree from the University of Electronic Science and Technology of China (UESTC), China. He was a visiting Ph.D. student at Politecnico di Torino, Turin, Italy, supported by the China Scholarship Council (CSC). His research interests include network observability, programmable networks, and cybersecurity.



Alessandro Cornacchia received the Ph.D. degree from the Politecnico di Torino, Italy. He is a Post-Doctoral Fellow with the King Abdullah University of Science and Technology (KAUST), Saudi Arabia. His research interests include multiple aspects of distributed systems and data center networks, with a special appreciation for monitoring, observability, and root-cause analysis. He served as a reviewer for ACM/IEEE journals and a PC member for ACM workshops.



Andrea Bianco (Senior Member, IEEE) received the Ph.D. degree in telecommunications from the Politecnico di Torino, Italy, in 1993. He is currently a Full Professor and the Vice Rector of the Internal Affairs, Politecnico di Torino. He has co-authored over 230 papers published in journals and presented at leading international conferences. His research interests include design of all-optical networks, switch architectures for high-speed networks and data centers, SDN, and networked music performance.



Idilio Drago received the Ph.D. degree from the University of Twente, The Netherlands. He is currently an Associate Professor with the University of Turin, Italy. His research interests include network security, AI, machine learning, and internet measurements. He is particularly interested in how AI and machine learning can help extract knowledge from network data and secure the network. He was awarded the IETF/IRTF Applied Networking Research Prize.



Paolo Giaccone (Senior Member, IEEE) is a Full Professor with the Department of Electronics and Telecommunications, Politecnico di Torino, Italy. He was with the Information Systems Networking Laboratory, Electrical Engineering Department, Stanford University, from 2000 to 2002. His research interests include the design of network algorithms, in particular for SDN networks and cloud computing systems.



Dingde Jiang (Member, IEEE) is currently a Professor with the School of Information and Communication Engineering, UESTC. His research interests include network measurement, modeling and optimization, network management, and network security, particularly in SDN, information-centric networking, energy-efficient networks, and cognitive networks. His research is supported by NSFC and the Program for New Century Excellent Talents with the University of Ministry of Education of China.



Marco Mellia (Fellow, IEEE) is currently a Full Professor with the Politecnico di Torino, Italy. He has co-authored over 300 papers published in journals and presented at leading conferences. His research interests include Internet monitoring, users' characterization, cyber security, and machine learning applied to different sectors. He won the IRTF ANR Prize at IETF-88 and the best paper awards at IEEE P2P'12, ACM CoNEXT'13, IEEE ICDCS'15, ACM CCR'16, and ITC'18. He is the Editor-in-Chief of the *Proceedings of the ACM on Networking*.